IFW AF/2176

CERTIFICATE OF MAILING
I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner of Patents and Trademarks, Mail Stop Non-Fee Amendment, P.O. Box 1450, Alexandria, VA 22313, on September 7, 2004.
Attorney for Applicant

PATENT
Docket No. ST9-99-145

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellant:	Shyh-Mei Ho et al.	)
Serial No.:	09/552,636	)
Filed:	April 19, 2000	) Group Art Unit: 2176
For:	**REPRESENTING IMS TRANSACTION DEFINITIONS AS XML DOCUMENTS**	)
Examiner:	Nathan Hillary	)

APPELLANT'S APPEAL BRIEF

Commissioner for Patents
Alexandria, VA 22313-1450

Sir:

On September 7, 2004, Appellant filed a timely Notice of Appeal from the Final Office Action mailed July 13, 2004. Appellant appeals from the rejection of all pending claims.

This Brief is being filed in triplicate under the provisions of 37 C.F.R. § 1.192. The filing fee set forth in 37 C.F.R. § 1.17(c) of Three Hundred and Thirty Dollars ($330.00) is now being submitted. The Commissioner is hereby authorized to charge payment of any additional fees associated with this communication, or to credit any overpayment, to Deposit Account No. 09-0460.

09/10/2004 HGUTEMA1 00000002 090460 09552636
01 FC:1402 330.00 DA

1. REAL PARTY IN INTEREST

The real party in interest is the assignee, International Business Machines Corporation, Armonk, New York.

2. RELATED APPEALS AND INTERFERENCES

There are no related appeals and interferences.

3. STATUS OF CLAIMS

The Final Office Action, mailed July 13, 2004, rejected claims 1-3, 7, 8-10, 14, 15-17, and 21 under 35 U.S.C. § 103 as being obvious over U.S. Patent Number 5,715,453 to Stewart (hereinafter Stewart), in view of a Research Disclosure publication 423111 from IBM (hereinafter "Research Disclosure"), and U.S. Patent No. 6,125,391 to Meltzer et al. (hereinafter "Meltzer"). Claims 4-6, 11-13, and 18-20 stand rejected under 35 U.S.C. §103(a) as obvious in view of Stewart, Research Disclosure, Meltzer, and U.S. Patent No. 6,038,393 to Lyengar et al. (hereinafter "Lyengar"), a publication by Brodsky entitled "XMI Open Application Interchange" (hereinafter "Brodsky"), and U.S. Patent No. 5,754,772 to Leaf et al. (hereinafter "Leaf"). Appellant appeals the rejection of pending claims 1-21.

4. STATUS OF AMENDMENTS

Appellant has not filed any Amendment subsequent to receipt of the Final Rejection.

5. SUMMARY OF INVENTION

By way of background, the present invention teaches interfacing a transaction processing system, such as an IMS (Information Management System), with newer computer systems. *Specification*, page 2, lines 15-16. In particular, the present invention teaches representation of IMS transaction definitions using an interchangeable format such as eXtensible Markup Language (XML). *Specification*, page 5, lines 17-18. The present invention enables compatibility between modern applications and reliable, stable, transaction processing systems, such as IMS through receiving IMS transaction definitions encoded in XML documents and generating Document Type Definitions (DTD) for use in encoding IMS transaction definitions in XML documents. *Specification*, page 5, line 20 – page 6, line 2, claims 1 and 7.

The present invention discloses these teachings through various embodiments of a system, method, and apparatus. In one embodiment, certain features are provided by an IMS gateway 50. *Specification*, page 12, line 20 – page 13, line 7, *Drawing* Figure 5. Specifically, the system 40 receives IMS transaction definitions 35 encoded in XML documents 44 from modern applications, such as a web browser 42. The XML documents 44 are processed by the IMS gateway 50 to produce the IMS-specific IMS transaction definitions 35 for use in IMS 10. *Drawing* Figure 5, *Specification*, page 12, line 5 – page 13, line 14. Similarly, the system 40 may be operated in reverse to convert IMS transaction definitions 35 into XML documents 44. *Id.*

Of particular note, the IMS transaction definitions 35 have a specific encoding and format required for interoperability with IMS 10. The IMS transaction definitions 35 further include macros that are understood exclusively by IMS 10. *Specification*, page 12, lines 14-16. These

macros such as a APPLCTN macro 36 and TRANSACT macro 38 and compatibility issues associated therewith are further described in the *Specification* on page 4, line 9 – page 5, line 9.

The present invention allows interfacing of a transaction processing system, such as IMS (Information Management System), with newer computer systems by properly converting the transaction definitions particular to the transaction processing system between a system-specific format and an open interchangeable format.

6. ISSUES

The following issues are presented for review:

I. Did the Examiner fail to establish *prima facie* obviousness of claims 1 – 21 where the limitations of the present invention are not found in the cited prior art?

II. Did the Examiner fail to establish *prima facie* obviousness of claims 1 – 21 where the Examiner failed to give proper weight to the limitation added by amendment further clarifying the transaction definition?

III. Did the Examiner fail to establish *prima facie* obviousness of claims 1 – 21 where the cited prior art references in combination as a whole, do not motivate or suggest the claimed invention?

IV. Did the Examiner fail to establish *prima facie* obviousness of claims 1 – 21 where the teaching to combine the prior art references is only found in the Appellant's disclosure using a hindsight combination?

7. GROUPING OF CLAIMS

The Examiner rejected Claims 1-3, 7, 8-10, 14, 15-17, and 21 as one group and Claims 4-6, 11-13, and 18-20 as a second group. Claims 1, 8, and 15 stand or fall as a group. Claims 2, 9, and 16 stand or fall together. Claims 3, 10, and 17 stand or fall together. Claims 4, 11, and 18 stand or fall together. Claims 5, 12, and 19 stand or fall together. Claims 6, 13, and 20 stand or fall together. Claims 7, 14, and 21 stand or fall together. Claims 4-6, 11-13, and 18-20 depend from independent claims 1, 8, or 15. Therefore, Claims 1-7 are representative of the remaining claims in each group and do not stand or fall together, and arguments why each representative claim is separately patentable are provided in the "Argument" section below.

8. ARGUMENT

I. Claims 1-21 are not obvious under 35 U.S.C. § 103 because Stewart, Research Disclosure, and Meltzer do not contain all the limitations of the present invention.

The Prior Art. The two references combined to reject the claims under Section 103 are summarized below.

Stewart. Stewart discloses a method that enables a web server to properly retrieve dynamic data results for web pages. Function calls within the web pages that query a data source for dynamic data are sent to one of a plurality of language processors as indicated by a

configuration file. *Stewart*, Abstract and Figure 1. The appropriate language processor receives the function call, interacts with the data source to obtain query results, and returns the query results to a transaction processor. *Id.* The transaction processor then puts the query results into a web page which is sent to a user. *Id.*

Thus, Stewart addresses the problem of required software changes in a transaction processor to accommodate dynamic data function calls that access different data sources or different types of data sources. *Stewart*, Col. 1, lines 59-63. Stewart allows the transaction processor to remain unchanged while suitable language processors are included to accommodate different types of dynamic data function calls.

In Stewart, rather than one or more language processors bundled within the transaction processor, the transaction processor appropriately directs different types of dynamic data function calls to separate language processors to improve flexibility and minimize software changes to the transaction processor. Thus, the invention taught by Stewart allows a web server to serve up web pages having dynamic data results from a plurality of data sources which may change without requiring a corresponding software change to the web server or its transaction processor.

Research Disclosure. Research Disclosure discloses conversion/translation between HTML files and XML files. *Research Disclosure,* page 1.

Meltzer. Meltzer teaches an interface between different communication networks and frameworks for Internet commerce. *Meltzer* Abstract, Col. 2, lines 32-41. Specifically, Meltzer teaches parsing of an XML document using a Document Type Definition (DTD).

Prima Facie Obviousness under 35 U.S.C. § 103 The Federal Circuit has held that "the 'subject matter' that must have been obvious to deny patentability under § 103 is the entirety of

the claimed invention," *Panduit Corp. v. Dennison Mfg. Co.*, 810 F.2d 1561, 1576 (Fed. Cir. 1987). In the present case however, Stewart, Research Disclosure, and Meltzer fail to disclose the entirety of the claimed invention. Specifically, Stewart, Research Disclosure, and Meltzer fail to disclose a transaction definition. This transaction definition is used throughout the independent claim elements. Consequently, Stewart, Research Disclosure, and Meltzer also fail to teach or disclose the remaining elements of the independent claims as they relate to a transaction definition.

Claim 1.

Stewart, Research Disclosure, and Meltzer fail to disclose "...receiving a document **comprising a transaction definition** encoded in XML, the **transaction definition** comprising **macro statements** that at least in part define a transaction...." as recited in claim 1 of the present invention. *Claims*, 1. Appellant has explained the importance of transaction definitions for interfacing with a transaction processing system such as IMS. Transaction definitions are defined as certain macros that provide all the information about structure and behavior of components for an IMS system. *Specification*, page 4, lines 9-13. The desirability of a method for encoding proprietary transaction definitions in an interchangeable format is also set forth. *Specification*, page 5, lines 14-16. For example, transaction definitions that include particular macros such as an APPLCNT macro and TRANSACT macro may enable the IMS to process arbitrary transactions such as ad hoc database queries. *Specification*, page 5, lines 4-5, 14-16. Appellants desire to emphasize a difference between a transaction definition which sets up the transaction processing system to process a transaction and the transaction itself.

The Examiner suggests that Stewart combined with the Research Disclosure discloses "receiving a document comprising a transaction definition encoded in XML and providing the decoded transaction definition to the transaction processing system." *Office Action*, June 13, 2004, page 3.

However, Stewart fails to disclose receipt of a document. The Examiner refers to a lengthy discussion in Stewart but still does not indicate what element in Stewart is considered the document recited in the claim. *Office Action*, June 13, 2004, page 2, referencing Stewart Col. 4, lines 12-36. "The examiner cannot sit mum, leaving the applicant to shoot arrows into the dark hoping to somehow hit a secret objection harbored by the examiner. [The examiner] must state clearly and specifically any objections. . . and give the applicant fair opportunity to meet those objections with evidence and argument." In re Oetiker, 24 U.S.P.Q.2d 1443, 1447 (Fed. Cir 1992)(Plager, J. concurring).

Appellant asserts that the Examiner is uncertain as to the applicability of Stewart to the Claimed invention. Consequently, rather than identify specifically what the Examiner considers to be the document, the transaction definition, or any other element of Claim 1, the Examiner is leaving the Appellant to guess what the Examiner considers representative of these elements. This practice is costly in terms of time and expense for the Appellant and contrary to the spirit of cooperation that should prevail in working with the Examiner. Consequently, with no specific evidence provided by the Examiner to the contrary, the Appellant asserts that there is also no document that is encoded in XML that comprises a transaction definition as recited in Claim 1.

Stewart

Stewart clearly teaches receipt of a web page request which is well known to be a message rather than a document. *Stewart* Figure 1. Stewart does disclose reading of a configuration file and processing of an HTML page. *Stewart* Col. 4, lines 25-28. However, Stewart fails to teach that the configuration file includes a transaction definition or that the configuration file is in XML. Furthermore, the configuration file fails to include "**macro statements** that at least in part define a transaction."

"[The] meaning of words used in a claim is not construed in a "lexicographical vacuum, but in the context of the specification and drawings." *Toro Co. v. White Consolidated Industries Inc.*, 199 F.3d 1295, 1301, 53 USPQ2d 1065, 1069 (Fed. Cir. 1999). MPEP §2106. According to the Manual of Patent Examining Procedure, "Office personnel must rely on the applicant's disclosure to properly determine the meaning of terms used in the claims." MPEP § 2106. *Markman v. Westview Instruments*, 52 F.3d 967, 980, 34 USPQ2d 1321, 1330 (Fed. Cir.) (*en banc*), *aff 'd*, U.S. , 116 S. Ct. 1384 (1996).

Appellant asserts that the disclosure clearly indicates that a transaction definition comprises macros. *Specification*, page 4, lines 9-13. A macro is "a shorthand representation for a number of lines of code." *See* The American Heritage® Dictionary of the English Language, Fourth Edition. Consequently, use of a macro requires that the system reading the macro understand how that macro expands to the larger number of lines of code. Appellant finds no teaching in Stewart that the configuration file includes macros or that the transaction processor reading the configuration file can understand or expand macros. The presence of macros in the transaction definition emphasizes that transaction definition define the transaction rather than

embodying the transaction itself. Furthermore, the presence of macros in a transaction definition presents a stark difference from any components in Stewart, the Research Disclosure, Meltzer, or the other art of record that might be characterized as a transaction definition.

As mentioned, Stewart does refer to processing of HTML pages. However, Stewart fails to teach that these HTML pages are documents that are received. Instead, the transaction processor in Stewart reads the HTML pages from sections stored in one or more macro files. *Stewart* Col. 4, lines 24-25. In other words, there is no document that is received. The HTML pages are pieced together using information in the macro files. *Stewart* Col. 6, lines 53-59. Furthermore, there is no teaching or suggestion in Stewart that the HTML pages include a "transaction definition" as that term is defined in view of the specification. Specifically, the HTML page is not defining a transaction. An HTML page is the **result of** a transaction initiated by a web page request from a user. The transaction is the request and subsequent retrieval of the web page. In Stewart, this transaction is defined by the Hyper-Text Transfer Protocol (HTTP) protocol used between the web server and the client browser.

Research Disclosure

Research Disclosure simply teaches translation or conversion between HTML and XML. Therefore, the Research Disclosure add nothing to overcome the deficiencies of Stewart.

Meltzer

Similarly, Meltzer fails to teach all the elements of Claim 1 or provide teachings lacking in Stewart. Specifically, Claim 1 recites "obtaining a Document Type Definition (DTD) specifying rules for decoding the **transaction definition**; parsing the XML document using the DTD to decode the **transaction definition**." Meltzer teaches parsing of an XML document

using a Document Type Definition (DTD). However, Meltzer fails to teach or disclose parsing a document comprising a transaction definition to decode the transaction definition.

Meltzer does teach decoding of a business interface definition. *Meltzer*, Col. 26, lines 21-25. However, a business interface definition is not a transaction definition as transaction definition is explained above. Meltzer describes a business interface definition as a "higher level document that acts as a schema used for designing interfaces that trade documents." *Meltzer*, Col. 11, lines 11-13. An example business interface definition is provided in Meltzer at Columns 12- 23.

Meltzer fails to teach or disclose that the business interface definition include macros. As indicated above, macros provide a shorthand way to easily represent a number of lines of code. In addition, the lack of macros in Meltzer can not be cured by the suggestion that macros simply be added. This is because macros require that the module reading the macro understand how to interpret and expand the macro into the appropriate number of lines of code.

Claim 1 further recites "providing the **decoded** transaction definition to the **transaction processing system**." This step provides that a transaction definition that is originally in a non-proprietary format such as XML is now in a suitable format for use by the transaction processing system. Consequently, the benefits of the claimed invention may be gained because a suitable transaction definition is provided such that ad hoc queries conforming to the transaction definition may be executed by the transaction processing system.

Again the Examiner asserts that Stewart teaches this element. *Office Action* page 3, referencing Stewart Col. 4, lines 12-36. This element states that the **decoded** transaction definition is provided to the transaction processing system. Stewart teaches a transaction

processor but it is not clear what the Examiner considers the "transaction processing system" in Stewart. Without any guidance from the Examiner, Appellant presumes the transaction processor is considered the transaction processing system.

Based on Figure 1 and the description in Stewart, Appellant identifies four potential elements that may be interpreted as decoded transaction definitions provided to the transaction processor: data results, web page requests, a configuration file, and macro files. Data results and web pages are clearly not decoded transaction definitions. The configuration file can not be a decoded transaction definition because, as configuration files are understood in the art, there is no decoding performed on them. Consequently, changing a configuration file through decoding would likely make the file unusable.

Finally, the macro files are not provided nor decoded. Macro files are not provided because the transaction processor access the macro files that are always available. Decoding means "to recognize and interpret (an electronic signal)." Merriam-Webster Online Dictionary, www.m-w.com. Therefore, the macro files are not decoded because the macro files are always in a format recognizable to the transaction processor of Stewart, there is no teaching to indicate otherwise.

Neither Stewart, Research Disclosure, nor Meltzer, singly or in combination, teach the entirety of the limitations of the present invention. Specifically, there is no teaching of a transaction definition. Furthermore, these references fail to teach or disclose all the operations that are performed in Claim 1 with respect to the transaction definition (i.e. decoding the transaction definition and providing the decoded transaction definition). Because these limitations render the present invention unobvious, Appellant asserts that claim 1 is allowable.

Claim 2.

Claim 2 depends from claim 1 and should be allowed for all the same reasons stated above regarding claim 1. In addition, claim 2 includes the limitation "...wherein the transaction definition comprises an APPLCTN macro." *Claims*, claim 2. Appellant submits that the claim term "APPLCTN macro" is used for a reason. "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). *MPEP* §2143.03. The term is not misspelled, not an acronym and not an abbreviation. Instead, the term is a name for a particular macro used in IMS systems. The term is not a trademark to the best of Appellant's knowledge and is well known to those of skill in the art. The term is clearly defined in the specification and in the provisional patent application no. 60/151,482 filed 8/30/199. *Specification* page 4, line 18 – page 5, line 3, pages 9 and Appendix A-4 of the document entitled "IMS Transactions in XMI" submitted as part of the above-identified provisional patent application. More particularly, the term is a term of art. Appellants are submitting with this brief two copies of definitions of these terms form an IMS transaction processing system manual to evidence that the term is a term of art.

Appellants respectfully submit that the Examiner failed to give proper weight and consideration to the term "APPLCTN macro" when examining the application. In support of the rejection, the Examiner cites to Stewart Col. 8, lines 56-66 which references function calls in the macro file. Based on the definition for macro set forth above, Appellants submit that a function call is not a macro. The portions of HTML code in the macro file of Stewart are macros because the HTML code is expanded by the dynamic data content which may be conditionally added. However, as explained above the HTML code is not a macro that defines a transaction definition.

Instead, the HTML code is the result of a web page request. Appellants find no reference to "APPLCTN macro" or its equivalent in the art references of record. Appellants submit that the Examiner's failure to give full weight to the term "APPLCTN macro" is reversible error.

Claim 3.

Claim 3 depends from claim 1 and should be allowed for all the same reasons stated above regarding claim 1. In addition, claim 3 includes the limitation "...wherein the transaction definition comprises an TRANSACT macro." *Claims*, claim 3. Appellants submit that for the same reasons as stated above with regard to the term "APPLCTN macro" the Examiner has failed to give proper weight to the term "TRANSACT macro." Appellants submit that this failure is reversible error as Claim 3 alone adds a patentable distinction over the prior art of record.

Claim 4.

Claim 4 depends from claim 1 and should be allowed for all the same reasons stated above regarding claim 1. In addition, claim 4 adds the limitation "...wherein the DTD comprises an XML Metadata Interchange (XMI) DTD..." *Claims*, 4. An XMI DTD is a very specific format of DTD produced by an XMI utility and described in the specification on page 15, line 3-10. An example of an XMI DTD is included in Appendix B of the application. An XMI DTD requires a specific set of software for reading and accessing the XMI DTD, specifically, the XMI document access classes which are used by the parsers. *Specification* page 15, lines 12-14.

The Examiner cites Lyengar to establish that business processes can be converted to Unified Modeling Language (UML). *Final* Office Action, page 8. Lyengar teaches a software development tool that works with UML. *See* Lyengar Abstract. Next, the Examiner cites Brodsky to establish that an UML can be converted into a DTD. Brodsky teaches interchange of

information using XMI. *See* Brodsky page 1. However, in Claim 1 the DTD comprises rules for decoding a transaction definition not an arbitrary business process as taught by Lyengar. In addition, Claim 4 recites a specific type of DTD an XMI DTD. Claim 4 does not recite conversion between UML and an XMI DTD. The limitations of claim 4 are thus not disclosed by the prior art. Appellant asserts that claim 4 is allowable.

Claim 5.

Claim 5 depends from claim 1 and should be allowed for all the same reasons stated above regarding claim 1. In addition, claim 5 teaches that the document is received at a "transaction processing system gateway". *Claims*, 5. The Examiner cites Leaf which teaches transaction gateway clients. *Final* Office Action, page 9. However, Appellants assert that claim 5 relates to receipt of a document. As discussed above and set forth in Claim 1, this document includes a transaction definition and includes macros. Leaf teaches receipt of HTTP requests which are not documents. *See* Leaf Col. 2, lines 33-34. In addition, as discussed above, HTTP requests do not include macros defining a transaction definition. Appellant asserts that claim 5 is allowable because the limitations of claim 5 are not disclosed by Leaf as asserted by the Examiner.

Claim 6.

Claim 6 depends from claim 1 and should be allowed for all the same reasons stated above regarding claim 1. In addition, claim 6 adds the limitation that the transaction definition is modeled in UML and that the UML is processed using XMI to create a DTD. The Examiner cites Brodsky and Lyengar in support of the rejection. *Final* Office Action, page 8. Yet neither Brodsky nor Lyengar make reference to a transaction definition. As established above, a

transaction definition is not simply a business process it is a definition for a particular transaction specific to the transaction processing system. In addition, the transaction definition includes macros that facilitate defining the transaction. The limitations of claim 6 are thus not disclosed by the prior art. Appellant asserts that claim 6 is allowable.

Claim 7.

Claim 7 depends from claim 1 and includes the limitations "...obtaining the transaction definition; obtaining a Document Type Definition (DTD) specifying rules for encoding the transaction definition; and parsing the transaction definition with the DTD to encode the transaction definition in an XML document." *Claims*, claim 7. The Examiner cites Stewart and Research Disclosure in rejecting Claim 7. Appellant asserts that claim 7 is allowable because the limitations of claim 7 such as obtaining the transaction definition and parsing the transaction definition are not disclosed by Stewart or the Research Disclosure as asserted by the Examiner. In particular, because Stewart fails to teach a transaction definition, especially one that includes macro statements that define the transaction definition.

Claims 9-14 and 16-21.

As to Claims 9-14 and 16-21, these claims depend directly or indirectly from Independent Claims 1, 8, and 15, and should be allowed for the reasons discussed above. Appellant asserts that these claims are allowable.

II. Claims 1-21 are not obvious under 35 U.S.C. § 103 because the Examiner has failed to give proper weight to the limitation further clarifying the transaction definition.

Prima Facie Obviousness under 35 U.S.C. § 103 The Federal Circuit has held that "the 'subject matter' that must have been obvious to deny patentability under § 103 is the entirety of the claimed invention," *Panduit Corp. v. Dennison Mfg. Co.*, 810 F.2d 1561, 1576 (Fed. Cir. 1987). MPEP §2143.03.

In the present case, Appellant amended Claims 1, 8, and 15 to clarify the transaction definition. Specifically, the Appellant added the clause "...the transaction definition comprising macro statements that at least in part define a transaction." *Claims*, claim 1. Appellants respectfully assert that the Examiner failed to give proper consideration to the added clarifying phrase. Therefore, the Examiner failed to consider the "entirety of the claimed invention."

"All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). *MPEP* §2143.03. Appellant finds no mention of how or where one of the prior art references teach or disclose the claimed feature that the transaction include macro statements. As described above, such a distinction is significant as macro statements require that the transaction processing system include the capability to interpret, understand, and expand the macro statements.

Appellants respectfully assert that this oversight by the Examiner has cost the Appellant unnecessary delay and expense in procuring a patent for the claimed invention. As the features included in the amended statement bolster Appellant assertion that the independent Claims 1, 8, and 15 are nonobvious. Appellants respectfully request that the rejection be overruled.

III. Claims 1-21 are not obvious under 35 U.S.C. § 103 because there is no motivation or suggestion to combine the Stewart reference with the Research Disclosure and/or Meltzer, Lyengar, Brodsky, and Leaf references.

The prior art. The Stewart, Research Disclosure, Meltzer, Lyengar, Brodsky, and Leaf references combined to reject the claims under Section 103 are summarized above.

Prima Facie Obviousness. "It is insufficient that the prior art disclosed the components of the patented device, either separately or used in other combinations; there must be some teaching, suggestion, or incentive to make the combination made by the inventor." *Northern Telecom, Inc. v. Datapoint Corp.*, 908 F.2d 931, 934 (Fed. Cir. 1990) See e.g. *Interconnect Planning Corp. v. Feil,* 774 F.2d 1132, 1143, 227 USPQ 543, 551 (Fed.Cir.1985). To establish *prima facie* obviousness, there must be some suggestion or motivation to modify the reference or to combine reference teachings to arrive at the claimed invention. "The teaching or suggestion to make the claimed combination ... must be found in the prior art, not in applicant's disclosure." MPEP 2143, citing *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991). "The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination." See *MPEP 2143.01*, citing *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990).

Appellant has argued above that Stewart, Research Disclosure, Meltzer, Lyengar, Brodsky, and Leaf do not teach or disclose a transaction definition encoded in XML, especially, a transaction definition that includes macro statements that at least in part define the transaction definition." *See* Claim 1. However, even if Stewart, Research Disclosure, Meltzer, Lyengar, Brodsky, and Leaf did teach a transaction definition as recited in the claims, Appellant finds no

motivation in **the references** to combine Stewart, Research Disclosure, Meltzer, Lyengar, Brodsky, and Leaf or the desirability of such a combination to arrive at the present invention. The cited Stewart, Research Disclosure, Meltzer, Lyengar, Brodsky, and Leaf references teach certain claim elements not related to transaction definition, but fail to disclose, suggest, or motivate modifying the primary reference Stewart to arrive at the present invention ***as a whole***.

Claim 1.

The Examiner rejected claim 1 by picking and choosing isolated teachings from the Stewart, Research Disclosure, Meltzer, Lyengar, Brodsky, and Leaf references and pasting them together to recreate the claims. None of these references suggest nor teach the combination of the present invention, receiving a document comprising a transaction definition encoded in XML, the transaction definition including macro statements that define the transaction. *Claims*, Claim 1. Because the Examiner has shown no motivation or suggestion to combine the prior art references relied on in the rejection, Appellant asserts that claim 1 is allowable.

Claims 2-21.

As to claims 2-21, these claims depend directly or indirectly from Claim 1 or related Claims 5 and 15. As the Examiner has shown no motivation or suggestion to combine the references of Stewart, Research Disclosure, Meltzer, Lyengar, Brodsky, and Leaf, under the rationale discussed above, Appellant asserts that these claims are allowable.

IV. Claims 1-21 are not obvious under 35 U.S.C. § 103 because the teaching to combine the prior art references is only found in the Appellant's disclosure using a hindsight combination.

The prior art. The Stewart, Research Disclosure, Meltzer, Lyengar, Brodsky, and Leaf references combined to reject the claims under Section 103 are summarized above.

Prima Facie Obviousness. For the present invention to be obvious, the suggestion to make the invention's combination must be found in the prior art. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991). It is "impermissible to use the claims as a frame and the prior art references as a mosaic to piece together a facsimile of the claimed invention." *Uniroyal v. Rudkin-Wiley*, 5 USPQ2d 1434, 1438 (Fed. Cir. 1988) (citing *W. L. Gore & Associates v. Garlock, Inc.*, 220 USPQ 303, 312). Yet, that is what the Examiner has done, citing references with certain the elements of the present invention but lacking all limitations of the present invention or any suggestion or teaching for the combination.

Appellants respectfully assert that if the prior art of record so clearly demonstrates the obviousness of the claimed invention, a single reference would teach more than just one or two elements of the claimed invention. However, the formation of the combinations used in the rejections is indicative of impermissible hindsight analysis by the Examiner. The shear number of references used seems to indicate that the claim terms were used in a key word search of the prior art. For certain claims up to six different references are relied upon. Once a key word hit was found, there appears to be little analysis performed to determine the applicability of relevance of the reference. The large set of text in Stewart cited against "receiving the document...and providing the decoded transaction definition" without any further explanation as

to how which claim elements are represented by which elements in Stewart is indicative of hindsight keyword analysis in support of a 103 obviousness rejection. Appellants respectfully assert that because such analysis is improper the rejections should be overturned.

Claim 1.

Appellant respectfully asserts that the teaching or suggestion to make the claimed combination is only found in the Appellant's disclosure. Neither Stewart nor Research Disclosure teach a transaction definition encoded in XML that comprises macro statements. The present invention provides the only suggestion for transaction definitions comprising macro statements. Neither Stewart, the Research Disclosure, or other references cited. Appellant asserts that claim 1 is allowable because the teaching to process a transaction definition encoded in XML that comprises macro statements is only found in the present invention and not in the prior art relied on in the rejection.

Claims 2-21.

As to claims 2-21, these claims depend directly or indirectly from Claims 1, 8, and 15. Appellant asserts that these claims are allowable as only the present invention teaches to combine the prior art references in the claimed invention.

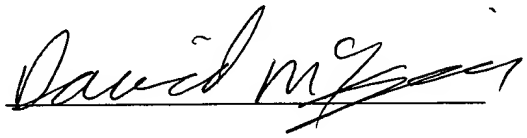
No Prima Facie Obviousness Established.

In view of the foregoing, the Examiner has not properly established *prima facie* obviousness of claims 1-21. Appellant respectfully requests reversal of the Section 103 rejection and allowance of claims 1-21. Appellant submits that the foregoing arguments further establish the non-obviousness of the present invention. Reversal of the rejections and allowance of the pending claims is respectfully requested.

SUMMARY

In view of the foregoing, each of the claims on appeal has been improperly rejected. Reversal of the Examiner's rejection and allowance of the pending claims 1-21 is respectfully requested.

Respectfully submitted,



David J. McKenzie
Reg. No. 46,919
Attorney for Applicant

Date: September 7, 2004
8 East Broadway, Suite 600
Salt Lake City, UT 84111
Telephone (801) 994-4646
Fax (801) 322-1054

9. APPENDIX

Claims involved in the appeal

1. (Previously Amended) A method for communicating with a transaction processing system using eXtensible Markup Language (XML) documents, the method comprising:

receiving a document comprising a transaction definition encoded in XML, the transaction definition comprising macro statements that at least in part define a transaction;

obtaining a Document Type Definition (DTD) specifying rules for decoding the transaction definition;

parsing the XML document using the DTD to decode the transaction definition; and

providing the decoded transaction definition to the transaction processing system.

2. (Previously Amended) The method of claim 1, wherein the transaction definition comprises an APPLCTN macro.

3. (Previously Amended) The method of claim 1, wherein the transaction definition comprises a TRANSACT macro.

4. (Original) The method of claim 1, wherein the DTD comprises an XML Metadata Interchange (XMI) DTD.

5. (Previously Amended) The method of claim 1, wherein the receiving step comprises receiving the document at transaction processing system gateway.

6. (Previously Amended) The method of claim 1, wherein the obtaining step comprises:

modeling an transaction definition in a Universal Modeling Language (UML) to produce a UML object model;

processing the UML object model using an XML Metadata Interchange (XMI) utility to create the DTD.

7. (Previously Amended) The method of claim 1, further comprising:

obtaining the transaction definition;

obtaining a Document Type Definition (DTD) specifying rules for encoding the transaction definition; and

parsing the transaction definition with the DTD to encode the transaction definition in an XML document.

8. (Previously Amended) A system for communicating with a transaction processing system using eXtensible Markup Language (XML) documents, the system comprising:

a document reception module configured to receive a document comprising a transaction definition encoded in XML, the transaction definition comprising macro statements that at least in part define a transaction;

a parser configured to obtain a Document Type Definition (DTD) specifying rules for decoding the transaction definition and further configured to parse the XML document using the DTD to decode the transaction definition; and

a transaction processing interface module configured to provide the decoded transaction definition to the transaction processing system.

9. (Previously Amended) The system of claim 8, wherein the transaction definition comprises an APPLCTN macro.

10. (Previously Amended) The system of claim 8, wherein the transaction definition comprises a TRANSACT macro.

11. (Original) The system of claim 8, wherein the DTD comprises an XML Metadata Interchange (XMI) DTD.

12. (Previously Amended) The system of claim 8, wherein the document reception module comprises a transaction processing system gateway.

13. (Previously Amended) The system of claim 8, further comprising:

a modeling tool configured to model a transaction definition in a Universal Modeling Language (UML) to produce a UML object model; and

an XML Metadata Interchange (XMI) utility configured to process the UML object model to create the DTD.

14. (Previously Amended) The system of claim 8, wherein the parser is further configured to obtain a DTD specifying rules for encoding a transaction definition as an XML document and to parse the transaction definition using the DTD.

15. (Previously Amended) An article of manufacture comprising a program storage medium readable by a processor and embodying one or more instructions executable by the processor to perform a method for communicating with a transaction processing system using eXtensible Markup Language (XML) documents, the method comprising:

receiving a document comprising transaction definition encoded in XML, the transaction definition comprising macro statements that at least in part define a transaction;

obtaining a Document Type Definition (DTD) specifying rules for decoding the transaction definition;

parsing the XML document using the DTD to decode the transaction definition; and

providing the decoded transaction definition to the transaction processing system.

16. (Previously Amended) The article of manufacture of claim 15, wherein the transaction definition comprises an APPLCTN macro.

17. (Previously Amended) The article of manufacture of claim 15, wherein the transaction definition comprises a TRANSACT macro.

18. (Original) The article of manufacture of claim 15, wherein the DTD comprises an

XML Metadata Interchange (XMI) DTD.

19. (Previously Amended) The article of manufacture of claim 15, wherein the receiving step comprises receiving the document at a transaction processing system gateway.

20. (Previously Amended) The article of manufacture of claim 15, wherein the obtaining step comprises:

modeling the transaction definition in a Universal Modeling Language (UML) to produce a UML object model;

processing the UML object model using an XML Metadata Interchange (XMI) utility to create the DTD.

21. (Previously Amended) The article of manufacture of claim 15, the method further comprising:

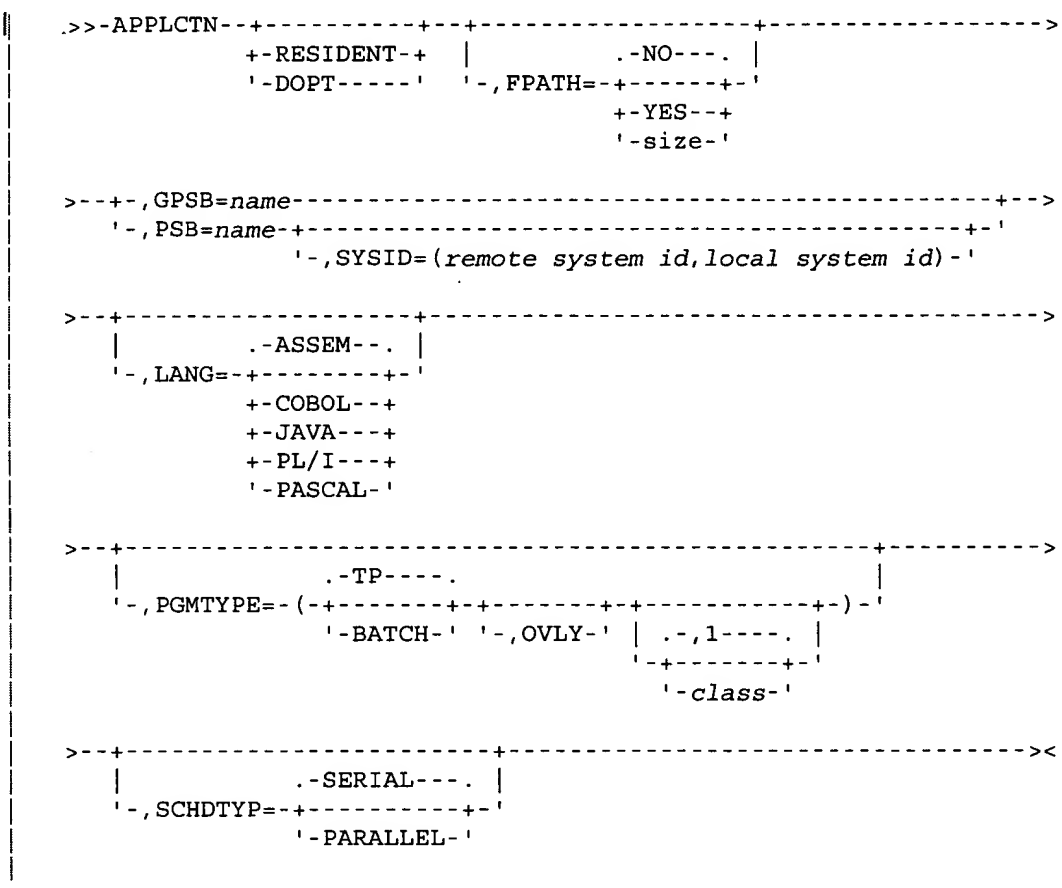
obtaining the transaction definition;

obtaining a Document Type Definition (DTD) specifying rules for encoding the transaction definition; and

parsing the transaction definition with the DTD to encode the IMS transaction definition in an XML document.

APPLCTN Macro

The APPLCTN macro allows you to define the program resource requirements for application programs that run under the control of the IMS DB/DC environment, as well as for applications that access databases through DBCTL. An APPLCTN macro combined with one or more TRANSACT macros defines the scheduling and resource requirements for an application program. Using the APPLCTN macro, you only describe programs that operate in message processing regions, Fast Path message-driven program regions, batch message processing regions, or CCTL threads. You do use the APPLCTN macro to describe application programs that operate in batch processing regions. When defining an IMS data communication system, at least one APPLCTN macro is required.



Positional Parameters

The positional parameter RESIDENT specifies that the PSB associated with this application program is to be made resident during system initialization. RESIDENT and DOPT (dynamic PSB option) are mutually exclusive. DOPT and SCHDTYP=PARALLEL are also mutually exclusive.

DOPT specifies that the PSB associated with this application program is to be located dynamically. If DOPT is specified, the following actions are taken during the execution of the IMS control region:

- Initialization does not perform a BLDL on the PSB associated with this application program. Thus the PSB does not need to be in any data set defined by the ACBLIB DD statement until it is actually required to process a transaction. This is not true, however, of the DBDs that the PSB might reference. All DBDs that are to be used online must be available during initialization in a data set defined by the ACBLIB DD statement. PSBs referencing DBDs that are missing during initialization are not scheduled.
- Each time the program associated with this PSB is scheduled, a BLDL is performed, and the latest copy of the PSB is located. A BLDL is not performed for associated DBDs, so the DBD cannot be modified until the system is reinitialized or until online change is used.
- When the program terminates, the PSB is deleted from the PSB pool as part of the termination process.

Neither RESIDENT nor DOPT is the default parameter. Rather, if neither RESIDENT nor DOPT is selected on the APPLCTN macro statement, IMS system initialization causes a BLDL to be performed on the PSB associated with the application being defined. The PSB is not made resident (that is, loaded from the ACBLIB) until the application is scheduled.

To provide greater flexibility in the user's control of the library containing the dynamic PSBs, the following restrictions are imposed:

- The PSB must reside in a library other than the primary ACBLIB and must be concatenated to it.
- The concatenated library containing the PSB must be in ACBLIB format.
- The IMS.PSBLIB data set cannot be used.

If the BLDL performed at the scheduling of the PSB determines that the PSB resides in the first concatenation of the IMSACB DD statement set, the PSB is stopped and an error message is sent to the master terminal. The PSB is not scheduled. If the dynamic PSB is added to the concatenation of the IMS.ACBLIB, causing the data set to be expanded to a secondary extent, that PSB is not available to the online system until the ACBLIB is closed and reopened by IMS.

For MSC remote applications, the APPLCTN macro provides documentation and a reference for its transactions. It does not generate a control block. The PSB need not be generated in systems where it is used as a remote reference. The PSB name is defined here for documentation purposes.

To dynamically reassign a transaction from remote to local processing, a control block must be present in the local system.

Related Reading: See the description of the `/MSASSIGN` command in *IMS Version 8: Command Reference* for additional information on dynamic reassignment.

Thus, if dynamic reassignment is desired, a transaction must be defined as local. That is, it must be under a local application having the same PSB name as the remote application (using the APPLCTN macro without a SYSID parameter). If dynamic reassignment is not desired, local definition is unnecessary.

Online changes to both the RESIDENT and DOPT options can be made via a MODBLKS system definition. However, PSBs associated with new application programs defined as RESIDENT (that is, changed from NONRESIDENT or DOPT to RESIDENT) during a MODBLKS system definition are not made resident until the next restart of IMS. Until that time, they are be treated as nonresident. Online changes made to already-resident PSBs (via an ACBGEN) cause the PSB to be treated as nonresident until the next IMS restart.

MPPs scheduled against dynamic PSBs are not allowed to go through the quick reschedule process or become pseudo WFIs.

Related Reading: For additional information on PSBs defined with DOPT see *IMS Version 8: Utilities Reference: System* under "Application Control Blocks (ACB) Maintenance Utility."

Keyword Parameters

To find which parameters apply to your IMS configuration, refer to Table 12.

FPATH=

Specifies whether (YES) or not (NO) this is a Fast Path-exclusive application program. FPATH=size, which determines the EMH buffer size required to run the transaction, overrides the EMHL execution parameter and implies FPATH=YES. The minimum specification for FPATH=size is 12; the maximum is 30720. FPATH=YES implicitly defines a wait-for-input (WFI) application program. The PGMTYPE= parameters that define the overlay structure and class are invalid if FPATH=YES is specified. The SYSID= parameter is also invalid if FPATH is specified.

If FPATH=YES is specified during a MODBLKS system definition, Fast Path must have been previously defined for the online system to which this change is made.

Fast Path-potential transactions must be able to run under two applications. One of these applications must be defined with FPATH=YES; the other with FPATH=NO. The application with FPATH=YES must also be defined with either the same transaction or with a routing code that can be assigned by the user Input Edit/Routing exit routine.

FPATH=NO must be specified if specifying LANG=JAVA.

Related Reading: See the TRANSACT macro in this information, and also the *IMS Version 8: Customization Guide* for information about the User Input Edit/Routing exit routine.

When Fast Path is included during system definition, the FPBUF parameter on the TERMINAL macro statement is ignored, except to determine the default EMH buffer size. Fast Path buffers are provided by the EMHB pool, which expands and contracts dynamically depending on the number of ETO terminals concurrently entering Fast Path transactions.

GPSB=

Causes the scheduling process of all environments to generate a PSB containing an I/O PCB and an alternate modifiable PCB. With

the GPSB= keyword, you do not need to perform the PSBGEN and ACBGEN, thus eliminating I/O to the ACBLIB.

GPSB= generates an I/O PCB named IOPCBbbb. The modifiable, alternate PCB is named TPPCBbb. With an alternate modifiable PCB, an application can use the CHNG call to change the output destination and send output to a destination other than the input destination.

You can make an online change to add the GPSB option to an existing application, or add a new application with the GPSB options by using a MODBLKS system definition. However, the GPSB option does not take effect unless the ACBLIB is also changed with online change.

If LANG=JAVA is specified and the Java(TM) application is a message processing program, the name specified on GPSB= is the name of the Java application class.

LANG=

Defines the language interface of the application program. You can use the following values with the LANG= keyword:

ASSEM is the default value.

If GPSB= is specified, LANG= is used to determine the language; it is used only with GPSB=.

If LANG=JAVA is specified, then FPATH=NO must also be specified.

PGMTYPE=

Specifies application program characteristics.

The first parameter describes the type of application program being defined. The default, TP, specifies that IMS schedules the program when messages processed by the program exist in the system. A program defined as BATCH can use DL/I in the IMS control program system region and can refer to the message queues. If BATCH is coded, all TRANSACT macro statements that follow are assigned normal and limit priority values of zero.

If a program is changed from batch to online using the online change facility, you must enter the `/ASSIGN` command to assign nonzero current, limit, and normal priorities (CPRI, NPRI, and LPRI keywords) to the transactions using that program. This is because the online change facility does not alter attributes that are changeable through the `/ASSIGN` command. Regardless of whether you specify new values for the transaction in the MODBLKS system definition, they are ignored during `/MODIFY` processing.

If FPATH=YES is specified

- TP specifies a message-driven Fast Path application program.
- BATCH cannot be specified. If BATCH is specified, an error message is issued. Fast Path nonmessage-driven application programs are not supported, and should be changed to run as BMPs.
- The PGMTYPE= parameters that define the overlay structure and class are invalid.
- The SYSID= parameter is invalid.

The OVLY parameter on the APPLCTN macro statement is no longer used by the system but is retained for compatibility. An execution time parameter, OVLA on procedure DFSMPR, is available when starting an MPP to indicate whether the overlay supervisor should be preloaded by IMS. For information on coding DFSMPR, see DFSMPR Procedure.

The third parameter of the PGMTYPE= keyword specifies the class to which the transaction codes specified in the following TRANSACT macro statements are to be assigned. This parameter must be a decimal number from 1 to 255. This value must not exceed the value given (by specification or default) on the MAXCLAS= keyword of the IMSCTRL macro. The default is 1. If the transaction code class is to be specified in the individual TRANSACT macro statements, this parameter need not be coded. If the transaction code class is specified in both the APPLCTN and TRANSACT macro statements, the APPLCTN macro specification is ignored, and the TRANSACT macro specification is used.

If the PGMTYPE= (,,class) parameter is to be changed online, the class value that is specified cannot exceed the definition (by specification or default) on the MAXCLAS= keyword of the IMSCTRL macro in the online system to which this change is to be made.

The numeric class subparameter must not be specified if FPATH=YES is specified.

PSB=

Specifies the name of the PSB associated with this application program definition. Each local PSB name must be unique. A remote

and a local application can each be defined as having the same PSB name. This is required in order to dynamically reassign a transaction from remote to local processing. The first character of the PSB name must be a letter. If PGMTYPE=TP, the PSB name must also be the program name.

Related Reading: For the description of the /MSASSIGN command, see *IMS Version 8: Command Reference*.

SYSID=

Specifies in the multiple-IMS system configuration, the system identification (SYSID) of the remote system (that system on which the application executes) and the SYSID of the local system (the originating system to which the responses are returned). The values specified must be numbers in the range from 1 to 2036.

The remote SYSID specified must also be defined in an MSNAME macro statement, but the local SYSID can be defined in any or all of the MSNAME, TRANSACT, and APPLCTN macro statements. If SYSID is specified, all other keywords except PSB are ignored.

If the SYSID parameter is specified in the APPLCTN macro statement, you need not specify the SYSID in the TRANSACT macro statement. If the SYSID is specified in both the APPLCTN and TRANSACT macro statements, the APPLCTN specification is ignored.

The SYSID parameter is independent of the link type (BSC, CTC, MTM, VTAM) specified on the TYPE= keyword of the MSPLINK macro statement.

Because the values associated with the SYSID= keyword can be changed using an /MSASSIGN command, you should not add SYSID= to an existing APPLCTN macro for a MODBLKS generation. Doing so causes the local application to be deleted after an online change sequence of commands.

A PSB cannot be initially defined or redefined from remote to local by using the SYSID= parameter on the APPLCTN macro during an online change.

The SYSID keyword parameter is invalid if FPATH=YES is specified.

SCHDTYP=

Specifies whether (PARALLEL) or not (SERIAL) this application program can be scheduled into more than one message region or batch message region simultaneously. The default value is SERIAL.

When the SCHDTYP= parameter is changed by a MODBLKS system definition from serial to parallel or vice versa, and the PSB, defined as resident, is not changed, the PSB is considered non-resident until the next IMS restart.

SCHDTYP= and the positional parameter DOPT are mutually exclusive.

TRANSACT Macro

The TRANSACT macro statement is used one or more times with each APPLCTN macro statement to identify transactions as IMS exclusive, IMS Fast Path potential, or IMS Fast Path exclusive. It specifies the transaction codes that cause the application program named in the preceding APPLCTN macro to be scheduled for execution in an IMS message processing region. It also provides the IMS control program with information that influences the application program scheduling algorithm. It can define a message editing routine.

If this macro statement is preceded by an APPLCTN macro defining it as remote (specifies SYSID), this transaction is generated as a remote transaction.

An IMS Fast Path-exclusive transaction is identified by a TRANSACT macro statement following an APPLCTN statement that specifies both PGMTYPE=TP and FPATH=YES.

TRANSACT macro statements that identify IMS Fast Path-exclusive transactions generate routing code entries. These entries have the same value as the corresponding transaction codes. They are used by the IMS Fast Path Expedited Message Handler routines to locate the Fast Path application program named in the preceding APPLCTN macro statement that processes the input message.

The SMU cannot define a Fast Path-exclusive transaction as able to issue commands.

An IMS Fast Path potential transaction is identified by a TRANSACT macro statement that specifies FPATH=YES following an APPLCTN statement that specifies PGMTYPE=TP and FPATH=NO.

An application defined as FPATH=NO cannot be run as a Fast Path region (IFP), even though it contains Fast Path potential transactions. Another application, defined with FPATH=YES, must exist and one of the following:

- A TRANSACT macro for the transaction
- A routing code defined by the RTCODE macro for the transaction

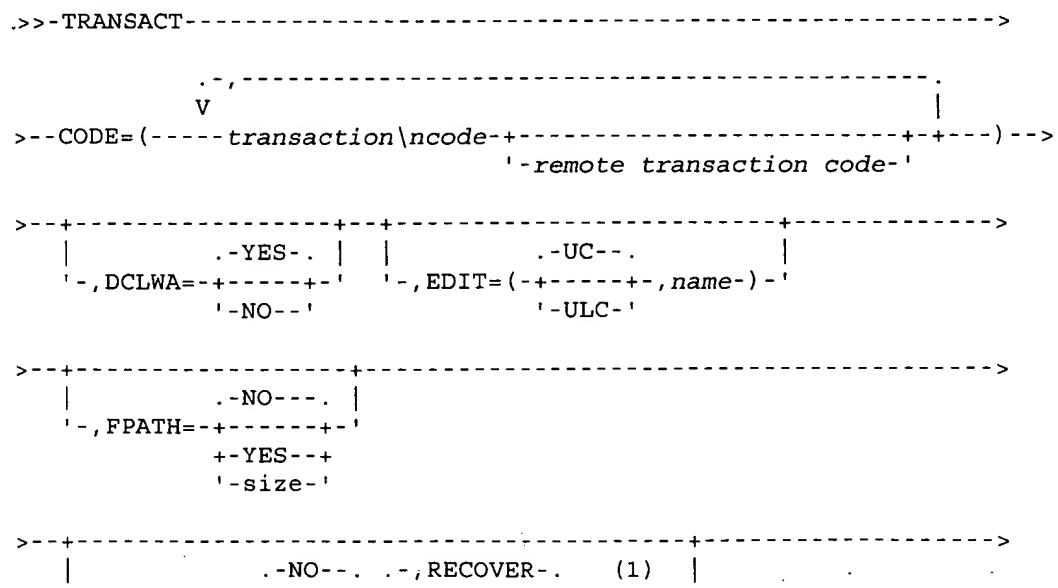
Related Reading: See the description of the Fast Path Routine for the Input Edit/Routing exit routine in the *IMS Version 8: Customization Guide*.

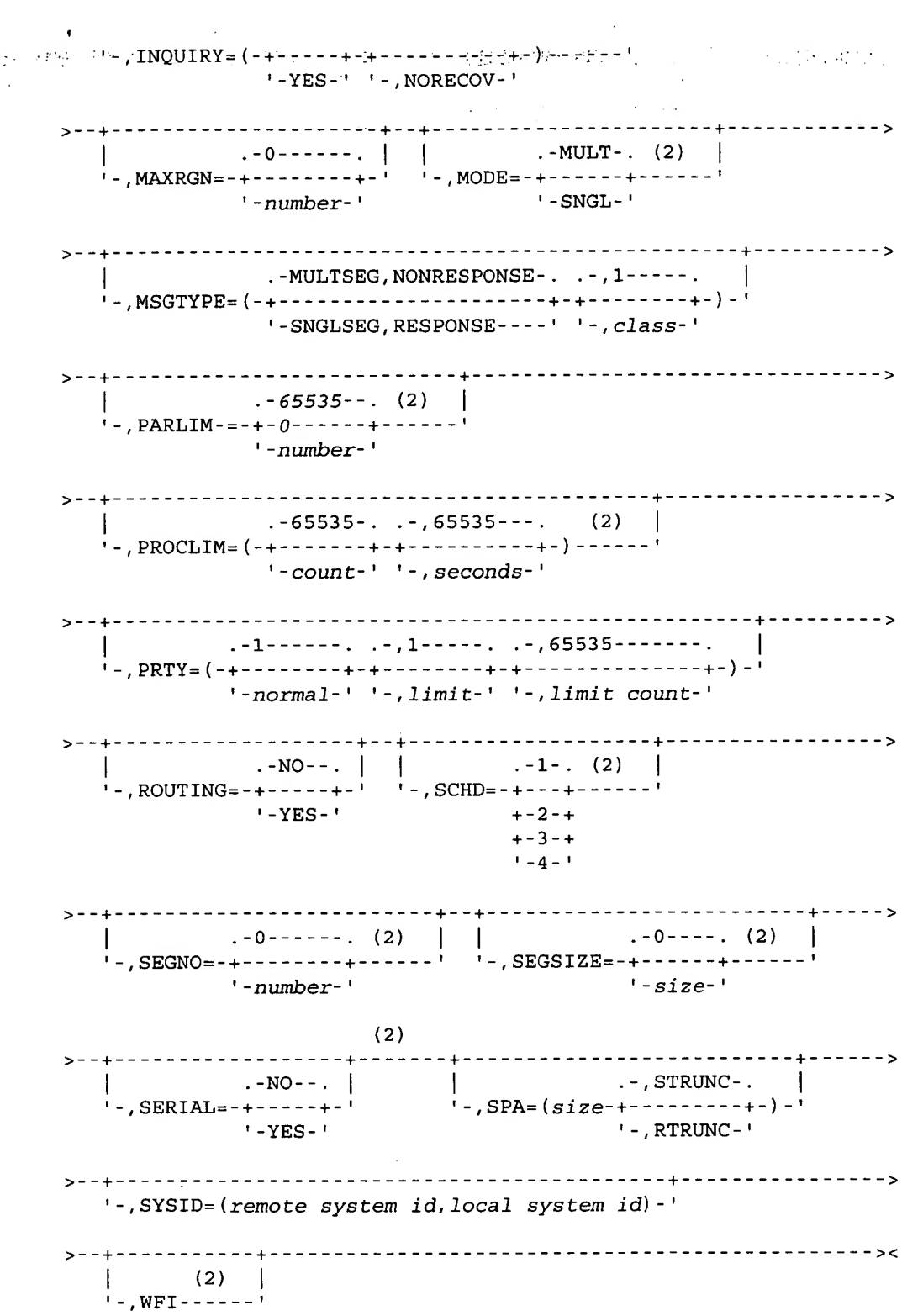
All Fast Path transactions are implicitly defined as recoverable.

MSGTYPE=(SNGLSEG,RESPONSE) must be specified on the TRANSACT statement for Fast Path potential transactions.

Additional routing codes can be associated with a Fast Path application program by using the RTCODE macro statement following the corresponding APPLCTN macro statement.

TRANSACT macro statements with FPATH=YES and RTCODE macro statements are invalid after APPLCTN macro statements that define an IMS batch message processing (BMP) application.





Notes:

1. NORECOV is not valid when NO is specified.
2. These operands are not appropriate if a transaction destined for processing in another (remote) system is being specified.

Keyword Parameters

To find which parameters apply to your IMS configuration refer to Table 12.

CODE=

Specifies one or more one- to eight-character alphanumeric transaction codes or remote transaction codes. If more than one code is specified, each transaction code name is assumed to have the same characteristics as all other keyword or positional parameter specifications or defaults. Enclose multiple names in parentheses and separate them with commas. Each character of transaction

codes and logical terminal names must be an alphanumeric character (A through Z, #, $, @, or 0 through 9). Transaction codes, logical terminals on the NAME macro, and linknames on MSNAME macros must comprise a set of values, each of which is unique in the system. That is, transaction codes, logical terminal names, and MSC linknames, collectively, cannot contain duplicates. The CODE operand is required.

Example:

```
CODE=(TRAN1,TRAN2,TRAN3)
```

DCLWA=

Specifies whether (YES) or not (NO) IMS should perform log write-ahead for recoverable, nonresponse mode input messages and transaction output messages. If not specified in the TRANSACT macro, the default is the DCLWA parameter in the IMSCTRL macro. The DCLWA parameter in the TRANSACT macro overrides the IMSCTRL macro parameter for this transaction.

Specify or accept a default of YES to ensure that both of the following occur:

- A nonresponse input transaction is made recoverable across IMS failures, prior to IMS acknowledging receipt of the input.
- Database changes are made recoverable prior to IMS sending associated output reply messages.

YES ensures that information in the log buffers is written to the IMS log, before the associated input acknowledgment or output reply is sent to the terminal.

Specify or accept a default of YES for all VTAM terminal types. For BTAM terminals, this option might increase input and output message path lengths.

Related Reading: For a information about the performance implications of selecting log write-ahead for transactions, see *IMS Version 8: Administration Guide: System*.

Specify NO if input message integrity and the consistency of output messages with associated database updates is not required. DCLWA does not apply to response mode or Fast Path input processing, and is ignored during IMS execution.

EDIT=

Specifies whether (UC) or not (ULC) the input data is to be translated to uppercase. The first parameter of this operand defines whether the transaction is uppercase/lowercase (ULC) as entered from the terminal, or if it is to be translated to uppercase (UC) before being presented to the processing program. The default is UC.

Specifying UC for VTAM terminals prevents the transmission of embedded device control characters.

You can also use EDIT to specify the one- to eight-character name of your own transaction input edit routine that edits messages prior to the program receiving the message. This name must begin with an alphabetic character. The specified edit routine (load module) must reside on the USERLIB data set prior to IMS system definition stage 2 execution. This routine cannot be the same as the one that is used on a LINEGRP or TYPE EDIT= parameter.

If FPATH=YES is specified, the EDIT= keyword parameter specifies whether (UC) or not (ULC) the transaction is to be translated to uppercase before being presented to the edit/routing exit routine. Specification of a user edit routine is valid on a Fast Path potential transaction; this specification is used when the transaction is routed to IMS. It is invalid on a Fast Path exclusive transaction.

For input from LU 6.2 devices, the user edit exit routine DFSLUEE0 is called instead of the transaction input edit routine specified in EDIT.

Related Reading: For more information about DFSLUEE0, see *IMS Version 8: Customization Guide*.

FPATH=

Specifies whether (YES, size) or not (NO) the transaction code is a potential candidate for Fast Path processing. FPATH=YES is effective only when specified on a TRANSACT statement that follows an APPLCTN statement that does **not** specify FPATH=YES; otherwise, the operand is ignored. FPATH=*size*, which determines the EMH buffer size required to run the transaction, overrides the EMHL execution parameter and implies FPATH=YES. The minimum specification for FPATH=*size* is 12; the maximum is 30720. The default is FPATH=NO.

Fast Path-potential transactions must be processed by a user edit/routing exit to determine whether the transaction is actually to be processed by IMS Fast Path. If it is to be processed by IMS Fast Path, the edit/routing exit routine associates the transaction with a

routing code. This routing code identifies which Fast Path application program is to process the transaction.

If a Fast Path-potential transaction is also defined with transaction command security by the Security Maintenance utility (SMU), that transaction must be processed by IMS, not Fast Path. (That is, an automated operator interface transaction can also be a Fast Path potential transaction if you program your user edit/routing exit routine to route the transaction to IMS and not Fast Path.) The SMU cannot define a Fast Path-exclusive transaction as able to issue commands.

If FPATH=YES is specified during a MODBLKS system definition, Fast Path must have been previously defined for the online system.

INQ= or

INQUIRY=

Specifies whether (YES) or not (NO) this is an inquiry transaction. The default is NO. You can specify this keyword either as INQ= or INQUIRY= If INQ= (or INQUIRY=) YES is specified, you can also specify whether (RECOVER) or not (NORECOV) this transaction should be recovered during an IMS emergency or normal restart. The specification of INQ=(NO,NORECOV) is invalid. The default is RECOVER.

For IMS Fast Path transactions, RECOVER must be specified.

INQ=YES should be specified only for those transactions that, when entered, do not cause a change in any database. Programs are prohibited from issuing ISRT, DLET, or REPL calls to a database when scheduled to process a transaction defined as INQ=YES. The INQ operands are not position dependent.

If the SPA parameter is specified (indicating that the transaction is conversational), INQ=(YES,NORECOV) cannot be specified.

Because switched terminals signed on for the INQUIRY LTERM, (one that has not specified /IAM) 2741 terminals are not allowed to enter transactions that update a database, the INQ=YES parameter must be specified for transactions that are entered by these terminals.

If an attempt is made to enter a transaction that is not specified as INQ=YES from one of these terminals, the transaction is rejected. A message indicating that the update transaction cannot be processed is sent to the terminal that entered the transaction.

MAXRGN=

Limits the number of message processing program (MPP) regions that can be concurrently scheduled to process a transaction. When the number of MPP regions is not limited, one transaction might monopolize all available regions.

If you specify zero, or if zero is defaulted to, no limit is imposed. The maximum number you can specify is 255.

If you specify SERIAL=YES or SCHDTYP=SERIAL in the APPLCTN macro, omit the MAXRGN parameter or set it equal to 0.

For non-0 values, you cannot specify MAXRGN= unless you also specify PARLIM=.

MODE=

Specifies that database buffers are to be written to direct access (flushed) upon each request for a new message (SNGL) by the processing program, or upon program termination (MULT). The default is MULT. Conversational and WFI transactions must be defined as SNGL. SNGL is forced for WFI applications.

This operand affects emergency restart. When MODE=SNGL, emergency restart only reprocesses the last completed message, regardless of whether one or more messages are scheduled and processed by a single load of an application program. Otherwise, emergency restart reprocesses those messages that were scheduled and processed by the single load of an application program, since the time of the previous checkpoint. The number of messages processed depends upon when the last checkpoint is issued.

The MODE= keyword parameters are checked on all IMS Fast Path-potential transactions for MODE=SNGL. If not specified, a warning diagnostic is issued.

MSGTYPE=

Specifies the type of transaction code (single or multiple segment), and whether the communication line from which the transaction is entered is to be held until a response is received. The MSGTYPE operands are not position dependent.

The transaction code can be single segment (SNGLSEG), or multiple segment (MULTSEG). It specifies the time at which an incoming message is considered complete and available to be routed to an application program for subsequent processing. The

defaults are (MULTSEG,NONRESPONSE,1).

If MSC-directed routing is used in a multiple IMS system configuration, IMS does not ensure that both the message and the transaction destined to process that message are either single segment or multiple segments.

The MSGTYPE= keyword parameters are checked on all IMS Fast Path-potential transactions for MSGTYPE= (SNGLSEG,RESPONSE). If not specified, a warning diagnostic is issued.

The first parameter of the MSGTYPE keyword specifies one of the following choices for number of segments:

MULTSEG

Specifies that the incoming message can be more than one segment in length. It is not eligible for scheduling to an application program until an end-of-message indication is received, or a complete message is created by MFS.

SNGLSEG

Specifies that the incoming message is one segment in length. It becomes eligible for scheduling when the terminal operator indicates end-of-segment.

The second parameter of the MSGTYPE keyword specifies one of the following response choices:

NONRESPONSE

Specifies that, for terminals specifying or accepting a default of OPTIONS=TRANRESP, input should not stop after this transaction is entered.

RESPONSE

Specifies that, for terminals specifying or accepting a default of OPTIONS=TRANRESP, no additional messages are to be allowed after this transaction is entered until this transaction sends a response message back to the terminal. Response mode can be forced or negated by individual terminal definition.

The third parameter of the MSGTYPE= keyword specifies the class to which this transaction code is to be assigned. This parameter must be a decimal number from 1 to 255. The default is 1.

Because the values associated with the class keyword can be changed using an `/ASSIGN` command, they will not be affected by an online change sequence of `/MODIFY` operator commands for an existing transaction, regardless of whether these values are altered during a MODBLKS system definition. The value specified must not exceed the MAXCLAS= value specified or accepted by default on the IMSCTRL macro statement. Regardless of specification made or default taken, remote transaction codes are assigned a class of zero. If the transaction code class is specified in the APPLCTN macro, this parameter need not be specified. If the transaction code class is specified in both the APPLCTN and TRANSACT macros, the APPLCTN macro specification is ignored for this transaction. Define CPI transactions with a different message class from that used for non-CPI transactions. IMS handles all CPI transactions as priority zero within the transaction class.

An input transaction from a 2741 line, or 2740 nonstation control is treated as a RESPONSE-type transaction, without regard to the type specified for this operand. Note also that MSGTYPE=RESPONSE is ignored during online processing for all terminals that do not operate in response mode.

PARLIM=

Specifies the threshold value to be used when SCHDTYP=PARALLEL is specified in the preceding APPLCTN macro instruction. If the current transaction enqueue count[6] exceeds the PARLIM value multiplied by the number of regions currently scheduled for this transaction, an additional region is scheduled upon a subsequent schedule event for this transaction's class. Events can include the following:

- Message Enqueue from terminal.
- Message insert from application. One region is allowed per syncpoint, EXPRESS=NO PCB. For example, when an application inserts two messages two messages for the same transaction code, only one region is scheduled at syncpoint.
- Assign command.
- Application term thread.

If PARLIM is not specified, the default value of NONE is assumed, and IMS allows the transaction to be scheduled in only one region at a time. Valid values for PARLIM are from 0 to 32767 and 65535, where 65535 disables transaction load balancing.

PARLIM=0 indicates that any input message can cause a new region to be scheduled because the scheduling condition is always be met (the number of messages will be greater than zero).

If you specify PARLIM=0, you should specify a MAXRGN value to limit the number of regions that can be scheduled to process a

particular transaction.

The value specified for PARLIM applies to message processing programs (MPPs) only; it is not supported for batch message processing programs (BMPs).

Because the value associated with the PARLIM= keyword can be changed using an /ASSIGN command, it is not affected by an online change sequence of operator commands for an existing transaction, regardless of whether it is altered during a MODBLKS system definition.

If you specify SERIAL=YES or SCHDTYP=SERIAL in the APPLCTN macro, omit the PARLIM parameter and accept the default.

PROCLIM=

Specifies the number of messages (count) of this transaction code a program can process in a single scheduling, and the amount of time (in seconds) allowable to process a single transaction (or message). Batch Message Programs (BMPs) are not affected by these settings.

The first parameter, *count*, can be changed using an /ASSIGN command. Therefore, its value is not affected by an online change sequence of operator commands for an existing transaction, regardless of whether it was altered during a MODBLKS system definition.

The *count* specifies the maximum number of messages sent to the application program by the IMS control program for processing without reloading the application program. The *count* value can range from 0 through 65535. If 0 is coded, the maximum number of messages sent to the application is one and the application program is reloaded before receiving a subsequent message. Code the *count* value at 65535 if no limit is to be placed upon the number of messages processed at a single program load. Values 1 through 65535 are eligible for quick reschedule processing. The defaults for PROCLIM are 65535 and 65535.

The *second* parameter specifies a numeric value, in seconds, which can range from 1 to 65535.

If Fast Path is used, *second* specifies, for a given transaction code, the amount of time (in hundredths of seconds) the program is allowed to process a single transaction message. A valid specification ranges from 1 to 65535; if 65535 is specified, no time limit is placed on the application program. The time represents real time that elapses during transaction processing (not accumulated task time). Real time is used because the input terminal is in response mode and cannot enter another transaction until the response is sent. The *count* subparameter is ignored.

The number specified by *second* represents the maximum CPU time allowed for each message to be processed in the message processing region.

The *count* value assigned is used to determine how many messages an application program is allowed to process in a single scheduling cycle. When the application program requests, and receives, the number of messages indicated in the *count* value, any subsequent requests result in one of two things.

1. IMS indicates "no more messages exist" if any of the following conditions are true:
   - The region is not an MPP.
   - The currently scheduled mode is not MODE=SNGL.
   - Equal or higher- priority transactions are enqueued for the region.

   IMS might, in fact, have other messages enqueued for the application program. It is the responsibility of the application program to terminate when it receives an indicator that no more messages are available. Termination of the application program makes the region it occupied available for rescheduling. This feature makes it possible for IMS to allow scheduling of higher-priority transactions that entered the system while the previous transactions were in process. In addition, if any equal-priority transactions are enqueued, they become eligible for scheduling on a first-in, first-out (FIFO) basis.

2. The region goes through quick reschedule and returns the next message to the application of all of the following conditions are true:
   - The region is an MPP.
   - The transaction is MODE=SNGL.
   - No higher priority transactions are enqueued.
   - Messages are still enqueued for the application.

The *seconds* value controls application program looping. You are not required to optimize the *seconds* value for program-transaction execution time. However, the *seconds* time value assigned should not be less than the expected per-transaction execution time. If the scheduled application program exceeds the product of *seconds* and *count*, the application program abends. If

an IMS STIMER value of 2 is specified on the DFSMPR macro, the region does not abend until completion of the DL/I call.

The application must not use MVS timer services, such as STIMER TASK, that override the IMS STIMER. IMS uses the IMS STIMER to time the execution of service message blocks. If an MVS TIMER is issued, it cancels out the IMS STIMER.

PRTY=

Specifies the values that determine the scheduling priority of this transaction. This priority also controls the priority of messages created by this transaction and sent to a destination in a remote system.

normal

The priority assigned to this transaction when the number of input transactions enqueued and waiting to be processed is less than the *limit count* value. The valid specification range is from 0 through 14. The default is 1.

limit

The priority to which this transaction is raised when the number of input transactions enqueued and waiting to be processed is equal to or greater than the *limit count* value. The valid specification range is from 0 through 14. The default is 1.

limit count

The number that, when compared to the number of input transactions queued and waiting to be processed, determines whether the *normal* or *limit* priority value is assigned to this transaction. The *limit count* value can range from 1 through 65535. The default is 65535.

When the limit priority is used, and the priority is raised to the specified limit priority value, the priority is not reduced to the normal priority until all messages enqueued for this transaction code are processed.

If you do not want the limit priority for this transaction, code equal values for the normal and limit priorities, and a limit count of 65535.

When a transaction is processed exclusively by a batch message program (BMP), code the normal and limit priorities as 0. The limit count value is ignored for a transaction processed by a BMP.

The APPLCTN macro statement forces the scheduling priority of all transaction codes associated with it to 0 if the program type is batch (PGMTYPE=BATCH on APPLCTN macro statement). However, a batch message processing region (BMP) can process transactions with scheduling priorities other than 0.

For remote transactions, the PRTY parameter determines the priority used to send the transaction to the processing system, which is termed the *MSC link message priority*. The three MSC link message priority groups are:

- Low
- Medium
- High

The low priority group consists of primary requests in the input terminal system. This group is assigned remote transaction priorities from 0 to 6. The medium group consists of secondary requests, responses, primary requests in an intermediate system, and primary requests in the input terminal system. This group is assigned a remote transaction priority of 7. The high group consists of primary requests in the input terminal system. Messages in this group are assigned remote transaction priorities from 8 to 14. Within each group, messages have a priority based on the current priority value of the transaction or remote transaction in the input terminal system for primary requests, and on the latest processing system for secondary requests and responses.

In an MSC configuration, the transaction priority determines the priority used to send messages inserted by this transaction across an MSC link. If the transaction inserts multiple messages to the same destination (for example, pages to a printer) and these messages must be sent in the order inserted, the normal and limit priority values should be the same. If the normal and limit priority values are not identical, messages inserted at a higher priority than previously inserted messages could arrive at their destination first. (This restriction does not apply to multiple segments of the same message.)

A transaction must have the same characteristics in all systems where it is defined. These characteristics include:

- Nonconversational/conversational
- SPA size if conversational
- Single-/multi-segment messages
- Noninquiry/inquiry
- Recoverable/nonrecoverable

ROUTING=

If MSC directed routing is used in a multiple IMS system configuration, specifies whether (YES) or not (NO) the application program

processing a transaction is informed of the system which originated the transaction.

If ROUTING=YES, an MSNAME corresponding to a logical path back to the originating system is placed in the I/O PCB. If ROUTING=NO, the name of the originating LTERM is placed in the I/O PCB. The default is NO.

SCHD=

Specifies the scheduling option used for other transactions when this transaction cannot be scheduled for internal reasons (database intent or no more space in PSB pool or DMB pool to bring in needed blocks). Values include:

1

Schedule only transactions of equal or higher priority in the selected class. This is the default.

2

Schedule higher-priority transactions in the selected class.

3

Schedule any transaction in the selected class.

4

Skip to the next class and attempt to schedule the highest-priority transaction in that class.

A loop counter in MPP scheduling breaks potential scheduling loops between SCHD=1 or SCHD=2 and SCHD=3 transactions. If scheduling fails due to intent conflicts for a SCHD=1 or SCHD=2 transaction, the counter increments and the next logical transaction is selected based on the SCHD=parameter of the failed transaction.

Each time the scheduler fails to schedule a transaction for intent, the counter is checked. If the count is greater than five, the next class of transaction is selected and the counter reset. This processing method can cause a delay in the current class.

For example, a long-running BMP has update intent on a database. Several transactions that have update intent on the same database with SCHD=1 are entered into the IMS system. Transactions in the same class that do not reference the database are also added to the system. However, the first group of transactions fails for intent, and none of the transactions is scheduled until the BMP terminates.

After each failed transaction, the loop counter increments. After the counter exceeds 5, the next class is scheduled, bypassing the transactions that do not reference the database and causing delays in their processing.

To avoid these delays, place the second group of transactions into a separate class, or run the BMP job at a different time.

SEGNO=

Specifies the maximum number of application program output segments that are allowed into the message queues per Get Unique (GU) call from the application program. It must be specified as a decimal number from 0 through 65535. The default is 0. If the default specification of 0 is used, the number of segments is not checked by the online system at execution time.

Because the value associated with the SEGNO= keyword can be changed using an `/ASSIGN` command, it is not be affected by an online change sequence of operator commands for an existing transaction, regardless of whether it is altered during a MODBLKS system definition.

SEGSIZE=

Specifies the maximum number of bytes allowed in any one output segment. It must be specified as a decimal number from 0 through 65535. The default is 0. If the default specification of 0 is used, the segment size is not checked by the online system at execution time.

Because the value associated with the SEGSIZE= keyword can be changed using an `/ASSIGN` command, it is not affected by an online change sequence of operator commands for an existing transaction, regardless of whether it is altered during a MODBLKS system definition.

The maximum output message segment to a LU 6.2 device is 32767. If a transaction is expected to send output to a LU 6.2 device, the SEGSIZE parameter should be no greater than 32767. However, this is not enforced during processing of the TRANSACT macro, because IMS cannot determine the device type for the message destination until output time.

SERIAL=

Forces serial processing of messages for a given transaction. When SERIAL=YES, U3303 pseudoabends do not cause the message to be placed on the suspend queue but rather on the front of the transaction message queue, and the transaction is stopped with a

USTOP.

The USTOP of the transaction is removed when the transaction or the class is started with a /START command.

The default for this keyword is NO, which means message processing is done as before with the messages placed on the suspend queue after a U3303 pseudoabend. Scheduling continues until repeated failures result in the transaction being stopped with a USTOP.

If you specify SERIAL=YES, PARLIM= and MAXRGN= must equal 0 (you can omit these parameters). If PARLIM= or MAXRGN= is not 0, SERIAL=YES is invalid, and the default of NO is assumed.

SPA=

Defines, by inclusion, that this transaction is a conversational transaction.

size

Specifies the size of the conversational scratchpad area (SPA). The size specified must be between 16 bytes and 32767 bytes inclusive.

STRUNC|RTRUNC

You can turn the truncated data option on (STRUNC) or off (RTRUNC).

If you specify SPA=STRUNC, IMS preserves all of the data in the SPA, even when a program switch is made to a transaction that is defined with a smaller SPA. The transaction with the smaller SPA does not see the truncated data, but when the transaction switches to a transaction with a larger SPA, the truncated data is used.

If you specify SPA=RTRUNC, the truncated data is not preserved.

STRUNC is the default.

When a conversation initially starts, and when a program switches, the STRUNC|RTRUNC option is checked and set or reset as specified. When the option is set, it remains set for the life of the conversation, or until a program switch occurs to a transaction that specifies the option is to be reset.

When a program switch occurs, the truncated data option for the new transaction is first checked, and if specified (either STRUNC or RTRUNC), that specification is set for the conversation and is used for the SPA inserted into the output message. If the option is not specified for the new transaction, the option currently in effect for the conversation is used.

Restriction: The CORE, DASD, and FIXED operands can no longer be used. If you specify them, assembly errors occur.

SYSID=

In the multiple-IMS system configuration, specifies the system identification (SYSID) of the remote system (the system on which the application executes) and the SYSID of the local system (the originating system to which the responses are returned). The values specified must be from 1 through 2036. The remote SYSID specified must also be defined in an MSNAME macro statement, but the local SYSID can be defined in any or all of the MSNAME, TRANSACT, and APPLCTN macro statements.

If the SYSID parameter is specified in the APPLCTN macro statement, you need not specify the SYSID in the TRANSACT macro statement. If the SYSID is specified in both the APPLCTN and the TRANSACT macro statements, the APPLCTN specification is ignored.

The SYSID parameter is independent of the link type (CTC, MTM, VTAM) specified on the TYPE= keyword of the MSPLINK macro statement. Because the values associated with the SYSID= keyword can be changed using an /MSASSIGN command, they are not affected by an online change sequence of operator commands, regardless of whether the values are altered during a MODBLKS system definition. The values will only take effect during an IMS cold start.

A Fast Path-exclusive transaction cannot have SYSID= specified. To assign a remote transaction as local, the associated APPLCTN macro must be defined as local. This means that no SYSID= parameter can be specified for the associated APPLCTN macro.

Restriction: A MODBLKS online change will be rejected if you attempt to add the SYSID parameter when a change to any other parameter specified on the TRANSACT macro is also requested

WFI

The positional parameter WFI specifies that this is a wait-for-input transaction. A message processing or batch processing application

program that processes WFI transactions is scheduled and invoked normally. If the transaction to be processed is defined as WFI, the program is allowed to remain in main storage after it has processed the available input messages. The QC status code (no more messages) is returned to the program if the PROCLIM *count* is reached; a command is entered to change the status of the scheduled transaction, database, program or class; the /DBR, /DBD or /STA commands relating to the databases used by the transaction are entered, or IMS is terminated with a checkpoint shutdown. MODE=SNGL is forced when WFI is specified.

---

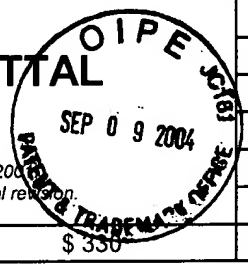
6. In a shared queues environment, the successful consecutive GU count is used instead of the enqueue count.

Please type a plus sign (+) inside this box → [+]

PTO/SB/05 (12/97)
Approved for use through 9/30/03. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL



Note: Effective October 1, 200[illegible]
Patent fees are subject to annual revision.

TOTAL AMOUNT OF PAYMENT	$ 330

Complete If Known	
Application Number	09/552,636
Filing Date	April 19, 2000
First Named Inventor	Shyh-Mei Ho
Group Art Unit	2176
Examiner Name	Nathan Hillary
Attorney Docket Number	ST9-99-145

METHOD OF PAYMENT (check one)

1. [X] The Commissioner is hereby authorized to charge indicated fees and credit any over payments to:

Deposit Account Number: 09-0460

Deposit Account Name: IBM CORPORATION

[X] Charge Any Additional Fee Required Under 37 CFR 1.16 and 1.17

[ ] Charge the Issue Fee In 37 CFR at the Mailing of the Notice of Allowance

2. [ ] Payment Enclosed:

[ ] Check [ ] Money Order [ ] Other

FEE CALCULATION

1. FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1001	770	2001	385	Utility filing fee	
1002	340	2002	170	Design filing fee	
1003	530	2003	265	Plant filing fee	
1004	770	2004	385	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	
SUBTOTAL (1)					$ 0

2. CLAIMS

		Extra		Fee from below		Fee Paid
Total Claims	-20 =		x	18	=	
Ind. Claims	- 3 =		x	86	=	
Multiple Dep. Claims		0	x	290	=	0

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	86	2201	43	Independent claims in excess of 3
1203	290	2203	145	Multiple dependent claim

SUBTOTAL (2) $ 0

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet	
1053	130	1053	130	Non-English specification	
1812	2520	1812	2520	For filing a request for reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1840*	1805	1840*	Requesting publication of SIR after Examiner action	
1251	110	2251	55	Extension for reply within first month	
1252	420	2252	210	Extension for reply within second month	
1253	950	2253	475	Extension for reply within third month	
1254	1480	2254	740	Extension for reply within fourth month	
1255	2010	2255	1005	Extension for reply within fifth month	
1401	330	2401	165	Notice of Appeal	
1402	330	2402	165	Filing a brief in support of an appeal	330
1403	290	2403	145	Request for oral hearing	
1451	1510	1451	1510	Petition to institute a public use proceeding	
1452	110	2452	55	Petition to revive - intentional	
1453	1330	2453	665	Petition to revive - unintentional	
1501	1330	2501	665	Utility issue fee	
1502	480	2502	240	Design issue fee	
1503	640	2503	320	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Petitions related to provisional applications	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	770	2809	385	Filing a submission after final rejection (37 CFR 1.129(a))	
1814	110	2814	55	Statutory disclaimer (37 CFR 1.120(d))	
				Other fee (specify) _____	

*Reduced by Basic Filing Fee

SUBTOTAL (3) $ 330

SUBMITTED BY				Complete (if applicable)	
Typed or Printed Name	David J. McKenzie of Kunzler & Associates			Reg. Number	46,919
Signature	[signature]	Date	Sep 7, 2004	Deposit Account User ID	

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner of Patents and Trademarks, Mail Stop Assignments, P.O. Box 1450, Alexandria, VA 22313-1450.